实验七　存储器

2020 年秋季学期

杨飞洋 191220138

目录

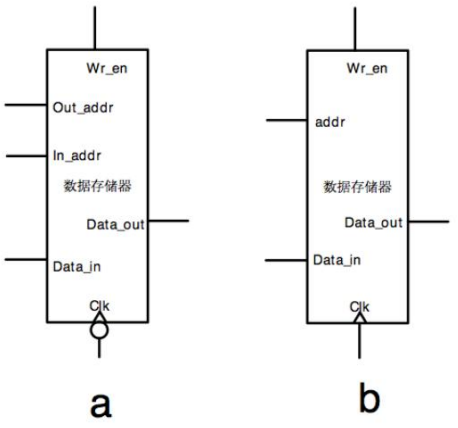
1. 实验目的：

了解 FPGA 的片上存储器的特性，分析存储器的工作时序 和结构，并学习如何设计存储器。

2. 实验原理（知识背景）

存储器是一组存储单元，用于在计算机中存储二进制的数据。存储器的端口包括输入端、输出端和控制端口。输入端口包括：读/写地址端口、数据输入端口等；输出端口一般指的是数据输出端口；控制端口包括时钟端和读/写控制端口。

存储器的工作过程如下：



FPGA 存储器的工作模式有很多，如：真双口 RAM、简单双口 RAM、单口 RAM、ROM 或者 FIFO 缓存等。

存储器模式	说明
单口存储器	某一时刻，只读或者只写
简单双口存储器模式	简单双口模式支持同时读写（一读一写）
混合宽度的简单双口存储器模式	读写使用不同的数据宽度的简单双口模式
真双口储存器模式	真双口模式支持任何组合的双口操作：两个读口、两个写口和两个不同时钟频率下的一读口一写口
混合宽度的真双口存储器模式	读写使用不同的数据宽度的真双口模式
ROM	工作于 ROM 模式，ROM 中的内容已经初始化
FIFO 缓冲器	可以实现单时钟或双时钟的 FIFO

3. 实验环境/器材

系统：windows10

开发软件：Quartus 17.1 Lite

开发板：DE10 Standard

仿真环境：ModelSim
芯片：Cyclone V ， 5CSXFC6D6

4. 设计思路

实验要求完成两个 16 * 8 的存储器，则考虑用两个二维数组来存储

```
reg [7:0] ram1 [15:0];
reg [7:0] ram2 [15:0];
```

第一个数组用.txt 文件来初始化，第二个数组用 IP 核.mif 文件来进行初始化。
.txt 文件内容：

```
@0 00
@1 01
@2 02
@3 03
@4 04
@5 05
@6 06
@7 07
@8 08
@9 09
@a 0a
@b 0b
@c 0c
@d 0d
@e 0e
@f 0f
```

Verilog 语言用.txt 文件初始化：

```
initial begin
    $readmemh("mem1.txt",ram1,0,15);
end
```

.mif 文件内容：

\dd	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	240	241	242	243	244	245	246	247	.....
8	248	249	250	251	252	253	254	255	.....

Verilog 语言用.mif 文件初始化：

```
(* ram_init_file = "test.mif" *) reg [7:0] ram2[15:0];
```

注意这里 ram2 是初次定义，不能提前定义 ram2

我把两个存储器放在一个模块里，是因为打算用一个选择端来选择执行哪一个存储器的功能，并且两个存储器共用输入输出端，这样就能省一些按键。

输入输出接口：

```
module ram1(chos,clk,we,addr,din,dout);
   input chos;
   input clk;
   input we;
   input [3:0]addr;
   input [7:0]din;
   output reg [7:0]dout;
```

chos 即选择端，clk 是时钟信号，we 是选择进行输入操作还是输出操作
addr 是地址，输入地址和输出地址共用，因为可以用 we 来区分，所以又省了 4 个按键
din 是输入的 8 位二进制数值，可以存储到存储器中，dout 是输出。
很自然的，逻辑代码如下：

```
always @(posedge clk)begin
   if(chos == 1)begin
      if(we)
         ram1[addr] <= din;
      else
         dout <= ram1[addr];
   end
   else begin
      if(we)
         ram2[addr] <= din;
      else
         dout <= ram2[addr];
   end
end
```

我加了一个 1s 的时钟信号，所以需要一个分频器，分频器模块：

```
module divider
#(
   parameter n
)
(
   input clk,
   output reg my_signal = 0
);
   integer clk_count = 0;

   always @ (posedge clk) begin
      if(clk_count == 25000000*n) begin
         clk_count <= 0;
         my_signal <= ~my_signal;
      end
      else
         clk_count <= clk_count + 1;
   end

endmodule
```

顶层文件中的调用：

```
wire signal;
divider#(1) d(CLOCK_50,signal);

assign LEDR[9] = signal;

ram1 s1(SW[9],signal,SW[8],KEY[3:0],SW[7:0],LEDR[7:0]);
```

用了 LEDR[9]来指示时钟信号。

5.实验步骤

一．利用 systembuilder 建立工程

选择 CLOCK、SWITCH、LEDR、button

二．编写 Verilog 代码

写一下相应模块的代码，在顶层文件中做相应的调用，进行编译调试，直至成功。

三．在板子上进行测试

6.测试方法

进行对存储器中数据的更改，并且输出验证效果。还要输出初始化的数值，保证初始化没有问题，利用 key 键作为时钟信号，验证输入/输出的工作时序。

7.实验结果

已验收。

8.实验中遇到的问题

1.一开始用的 1s 的时钟信号在验证输入/输出的工作时序方面不是非常友好，所以之后用了 key 键来模拟时钟信号，这样结果会很清楚。
2.一开始用 switch 键模拟时钟信号，会出现异常，是因为 switch 没有消抖，所以用 key 键模拟时钟信号。
3.在初始化存储器的时候，txt 文件读不出来，是因为编译器的问题，不接受绝对路径，接受同目录下的相对路径。

9.实验得到的启示

1．一定要学会用分模块编程，会让结构更加清晰，debug 更加容易
2．有问题可以向老师助教请教、和同学讨论、上网搜，很多时候是思路或结构出现了问题。

10.意见和建议

1.这种实验测试文件不好测试，希望老师可以直接说清楚不用测试文件，只需要在板子上测试。
2.希望给一些实验原理的指导，不然自己搞有点懵。
3.不知道开发板上其他键的作用是什么，总感觉现有的有些不够呢。